* **1**

采药问题，典型的0-1背包问题，非常具有典型性，是值得反复研读的题目！！

该问题可抽象为：

有一个容量为 V 的背包，和一些物品。这些物品分别有两个属性，体积 w 和价值 v，每种物品只有一个。要求用这个背包装下价值尽可能多的物品，求该最大价值，背包可以不被装满。每个物品都有两种可能的情况，即在背包中或者不存在（背包中有0个该物品或者1个），所以我们把这个问题称为 0-1 背包问题。

用dp[i][j]标识装前j种物品，时间不超过i的最大价值。若物品 i 被放入背包，设其体积为 w，价值为 v，则 dp[i][j] = dp[i-w][j-1] + v。即在总体积不超过 i-w 时前j-1件物品可组成的最大价值的基础上再加上j物品的价值 v；若物品不加入背包，则 dp[i][j] = dp[i][j-1]，即此时与总体积不超过 i 的前 j-1 件物品组成的价值最大值等价。

得到递推方程为 dp[i][j]=max(dp[i-w][j-1]+v, dp[i][j-1])

改进版：因为每次状态转移都是从前j-1种到前j种这个相邻状态之间转移，所以我们可以在相邻状态间变化时去除j这个一维的标识，于是二维数组可降维至一维数组，空间复杂度降低~

* **11**

水题！

* **1000**

水题。

* **1002**

方便记忆的电话号码，是一个字符串处理的问题，一个套路题型，需要做的便是先将输入的字符串标准化，关于大写字母的映射，可以额外编写一个字符数组作为映射，然后调用qsort()函数对于字符串排序，排序时调用了strcmp()函数，该函数如果字符串相等，返回0，如果不等，那么返回第一个不相等的字符的差值，刚好可以用于字符串排序操作。最后对于排好序的字符串数组遍历得到所有重复的字符串并打印出来~题目难度不大，是标准的模板题，但要熟悉写法。此外本道题还调用了cstring库的strlen()得到输入的字符串数目

* **1003**

水题，对于简单的数学公式计算，但要注意float类型转换！

* **1004**

水题，注意精度问题，此处选择double型数据，可以保证精度不损失，并且可表示的位数也很高~

* **1006**

最大连续子序列和问题，是一个经典的dp问题，给出一段序列，选出其中连续且非空的一段使得这段和最大。

令b[j]表示以位置 j 为终点的所有子区间中和最大的一个。如j为终点的最大子区间包含了位置j-1,则以j-1为终点的最大子区间必然包括在其中。

如果b[j-1] >0, 那么显然b[j] = b[j-1] + a[j]，用之前最大的一个加上a[j]即可，因为a[j]必须包含。

如果b[j-1]<=0,那么b[j] = a[j] ,因为既然最大，前面的负数必然不能使你更大。

它的程序的书写可以说是非常标准的模板，一定要熟悉！

* **1007**

DNA排序问题，题目难度不大，主要是关于逆序对的排序，本道题用到了结构体排序，调用了qsort()函数对其进行快速排序，感觉思路上比较清晰，是一个比较不错的程序的书写~

* **1011**

已知一堆切分后的木棍各个长度，请将它们拼成原有的统一的长度，要求统一的长度是可以拼出的长度中最短的，拼木棍问题，典型的深搜问题，但是难度较大。简单的深搜方程还可以勉勉强强码出来，但是最原始的深搜代码会带来TL，最后又用了几个强力的剪枝策略才AC..关于剪枝策略的思考及复现理解得不是很好 orz…

* **1017**

装箱问题，最主要的是对于6\*6的箱子划分好对于不同大小的箱子恰好装满时的各个箱子具体数目，贪心问题。

* **1064**

给了一堆线，每个线的长度也给了，需要确定一个最长的切割网线长度，按此长度切割后能达到指定数量网线。

本题乍一看毫无思路，后来有人提到了二分法，就是先假定最长的网线长度是一堆网线中最短的网线长度+1，接下来用二分法去不断逼近，如果到最后精度小于0.01的话仍然没有可行解，那么程序会退出返回0。关于二分法的使用是本题的一个特色之处，同时利用程序输入均为小数点后两位的特点，先进行乘100处理，之后二分法遍历得出最优解！！

本道题很有思维性，是二分法的一个很好的应用，要多多了解！

* **1083**

思路转换的题型，将挪动桌子的思路转换为经过过道的这一思路点！注意memset赋初值函数的用法，以及对于memset系统函数的调用需要导入string库。

* **1088**

最长路径，对于上下左右移动设置一个数值，对于二维层面的解答需用到二维矩阵去表示，其中最核心的是递归思想要有递归思想的概念。

* **1091**

难度较大，涉及到数学层面的知识：

[对于不全为 0 的非负整数 a、b，gcd（a，b）表示 a，b 的最大公约数，必然存在整数对 x，y ，使得 gcd（a，b）=ax+by ]

紧接着又有容斥原理的引入，同时还需要注意输入数据是否越界无法表示，关于容斥原理的实现利用到了DFS，不过此处的递归算法理解得不好，要多留意归纳！

* **1183**

反正切函数的应用，属于数学题，关键是分析数学公式，转换为因子式： (b-a)\*(c-a)=a^2-1 -->b+c=m+n+2\*a，其中m、n为因子，然后就是精度的转换，int型可能会溢出，采用long long型数制存储数据~

* **1251**

本道题是最小生成树模板题型，难度一般，一定要熟练掌握，对于最小生成树，最重要的是并查集模板程序的书写这一块！

* **1562**

给出图，找图的连通分量数目，是一个经典的问题，本处给出了用广搜加遍历的方法获取所有的连通数目，解法的思路很经典值得一看！！

* **1657**

引入国际象棋的规则，计算最短步数，最重要的是需要对于每种情况考虑清楚，涉及到基本的数学上的思考。

* **1644**

放苹果问题，是一个递归问题，主要是要自己能找到递归的条件：1. 至少空出一个盘子不放（相对于目前），则转化为在(n-1)个盘子里放m个苹果，在剩下的盘子里放入0个苹果，即apple(m,n-1)； 2. 所有盘子至少放入一个。先在每个盘子里都放入1个苹果，剩下m-n个苹果，再在n个盘子里进行放置。即apple(m-n,n)。 然后是结束的条件，有三种。 1. 当m==0时，苹果已经放完，只有一种情况，即不放置，返回1； 2. 当n==1时，只有一个盘子可以放置，返回1； 3. 当m<n时，苹果数小于盘子数，和将m个苹果放入m个盘子情况相同。

关于本道题的递归思路，我并没有想到，这种题型一定要多加理解，学会自己分析递归问题！

* **1700**

贪心问题，经典的过河问题，题目最重要的是要把题目分析清楚，把住题目的脉络，有固定的思路去解：首先要将所有过河时间按照升序排序，接下来：

1、一个人时，直接过河，t[0]

2、两个人时，过河由时间较长者决定 t[1]

3、三个人时，过河时间恒为 t[0]+t[1]+t[2]

4、>=4个人时，策略是每次将两个最慢的人送至彼岸，有两个策略：

a、 两小送两大：t[0]和t[1] 先去；t[0]回来；t[n-1]和t[n-2]去；t[1]回来

b、最快送两慢：t[0]和t[n-1]先去；t[0]回来；t[0]和t[n-2]再去；t[0]回来

由此，题目已经被拆解开来，不断地执行以上步骤，即可得出最优解，不过对于题目的理解尚存有问题，为什么过河问题就是这种思路的解法？

本道题用到了cstring库的memset()函数用于数组的初始化，algorithm库的qsort()函数用于数组的快速排序。

* **1833**

排列题，找到从当前序列开始 指定k个后的序列，采用了两种方法去解题，第一个方法是直接还原如何求下一排序的方法，理清了关系后，难度不大重要是掌握 qsort()快排函数以及相应的compare函数的书写；第二个方法是利用next\_permutation() C++内置函数直接求得下一个排序，简单无脑…并且使用了reverse()函数求出当序列到最后一个序列后再次全部逆序回到最初始的状态~

* **2000**

金币问题，算是道水题，就是简单的数学计算 推断出连续天数区间，用到了cmath库的ceil()上取整函数（还有个floor()下取整函数），以及sqrt()函数用于计算开方值。

* **2255**

给定一棵二叉树的前序遍历和中序遍历的结果，求其后序遍历。二叉树的遍历一般存有一定的套路，这种重建二叉树难点在于建树该如何建，是一道比较经典的题目，要细细研读！！

* **2690**

首字母大写，思路很水

但是用到了两个函数可以了解，一个是gets()函数，用于接收一行一行的输入的字符串，对于空白符不会结束；另一个是将字符大小写转换的函数 toupper()以及tolower()函数~

* **2691**

枚举求离散形式的极值点，难度较小，水题~

* **2692**

假币问题，通过已经称重得到的结果去判别哪个为假币，采用枚举的方法将每个硬币如果是假币时的各种情况列举出来，再逐个判别。此处犯了个逻辑性的错误，就是对于天平的一侧倾斜，比如右边高，那么枚举判别某个硬币是否较轻时，一定要判别该硬币是否在天平右侧！而不是硬币不能出现在天平左侧这个逻辑性的问题！本程序中用到了cstring库的strchr()用于判别字符串中是否有应查找的子串，若有返回地址，若无则返回NULL。

* **2694**

逆波兰表达式问题，递归思路，可以用二叉树的绘制去理解这个思路，从根节点到叶节点，回溯计算至根部。注意本道题用到了cmath库的atof()函数用于获取字符串的数值~本道题需要吐槽的是，用cout输出double型数值时因为精度不够，wa了一次，后改为printf()后才成功AC，也是蛮尴尬的…

* **2706**

麦森数求解，本道题实际上仍是求解2的幂

次，当然多了一些附加问题，比如求2的幂次的位数，利用了log 10()函数去求出2的幂次项的位数，当然是分析了2^N-1与2^N的个位肯定不会借位得到了2^N-1的位数，其中关于位数的求解，突然间让我想到了关于2的幂次项有多少位数可以用这种方法判定进而决定要开多大的数组去存储。关于log以10为底的函数的C++实现，调用了cmath库的log()函数，但是此处log()函数默认是以e为底，于是要求以10为底的形式需要进行换底公式运算。

本道题其余地方的解法仍是利用了快速幂+高精度乘法运算去求解，此处有一个数学上的关系，就是乘数与被乘数的乘积当只关注低500位时，那么乘数与被乘数只需要低位的500位相乘取低500位的数即可，这种关系通过纸上推演很容易了解，当然这个地方的理解很关键！本道题开了十万进制的乘积运算，即一个元素五位，利于最后的按格式打印，故数组也应被定义为long long型。在输出时 调用了iomanip库的setfill()函数以及setw()函数以对五位的数据进行高位补零操作。另外调用了cstring库的memset()函数对数组初始化，以及memcpy()函数对于数组进行复制操作。

* **2712**

计算细菌繁殖的数目，题目难度较小，需要注意的是数据的数制形式，此题应为long型。

* **2733**

判断闰年，水题。

* **2737**

大整数除法，算是高精度运算题目中难度较大的一类题型，有许多细节需要考虑，除数如何添0，每次添多少0合适，如何循环相减计数商的每个对应位的值，对于商高位的0如何避免打印出来，以及商为0的情况要分析，最核心的地方就是需要写出高精度减法运算。本道题中使用了 algorithm库中的reverse()函数，以及cstring库中的memset()及memcpy()函数，这几个函数使用的频率很高，需要留心它们的使用方法~

* **2742**

统计字符出现的最大次数，难度较简单，留意memset字符串函数的使用以及sizeof()函数和strlen()函数的区别。

* **2744**

字符串子串问题，题目难度不是很大，重要的是对于字符串的几个模板函数要熟悉使用：strstr 查找匹配的子字符串，strcpy 复制字符串，strncpy 复制子字符串，strlen 获得字符串的长度，algorithm库中的reverse()函数获取数组的逆序形式。

* **2745**

显示器问题，有七个段去表示数字，分别开七个数组去存储，然后采用模拟法去打印出来数型。在此题中，碰到的问题是：对于判别输入数是否均为0，与的反面为或，而不是与！在此处WA了几次，不该！

* **2746**

约瑟夫问题，难度不大，主要需要理清题目的脉络，把逻辑关系分析出来，用一个sign数组用于标识节点是否被删去，对于memset初始化数组的形式需要熟悉！

* **2748**

字符串元素的全排列题，思路与之前的元素排序题类似，学会调用合适的库去解决问题。本道题一开始是先对数组元素来个快排，接下来就调用 next\_permutation()函数逐个求出下一个序列，其中注意全排列的个数为Aii=i\*(i-1)\*….\*1；本道题调用了cstring库的strlen()函数求解字符串的长度，调用了algorithm库的qsort()函数对于字符串进行快排，以及next\_permutation()函数去求出字符串的下一序列~

* **2750**

鸡兔同笼问题，水题。

* **2754**

八皇后问题，很典型的深度优先搜索问题，加上约束条件共同判别。将八皇后的问题转换为八叉树结构：我们每取出一个皇后，放入一行，共有八种不同的放法，然后再放第二个皇后，同样如果不考虑规则，还是有八种放法。于是我们可以用一个八叉树来描述这个过程。从根节点开始，树每增加一层，便是多放一个皇后，直到第8层（根节点为0层），最后得到一个完全八叉树。其实就是用DFS去搜索八叉树，其中需要加上约束条件，就是行、列、斜线都没有别的皇后，转换为图的结构去考虑也就是：

abs(i-k)==abs(place[i]-place[k])||place[i]==place[k] 代表的是同列及同斜线出现别的皇后。接下来就是深搜，从根节点开始，逐层赋值元素，对应于八皇后每行选取列值，然后如果每次放置的皇后满足约束条件，那么进入下一层的搜索，回溯条件即为约束条件不满足时返回当前到达的树层，继续到另一个分叉去搜索；当搜索至最后一层后，将这一满足条件记录下来~

本道题只用到了abs函数去判别是否处于同一斜线这一约束条件，本道题运用的深搜思想很重要，代码可以多读几遍去体会其中的思想！

<https://blog.csdn.net/u014082714/article/details/44901575> 对应的博客讲解~

* **2756**

二叉树问题，冠上二叉树的噱头，其实只是二叉树的树形结构题，难度较小，就是除二递归~

* **2757**

最长递归子序列，算是比较简单的动态规划问题，就是每次都从前j-1个最大长度去比对，找出最大的长度，即为在第j位的最长子序列长度。最后为了得出总共序列的最长子序列的长度，需要遍历每一位得到的最长子序列长度找到最长的那条就好~

dp[1]=1

dp[i]=max(1,dp[j]+1|dp[i]>dp[j]&&i>j)

* **2760**

最大路径，数字三角形，动态规划问题，自底向上逆序找出每层结点当前最大值 直至找至最上面层时得到使数据最大的路径。

* **2787**

算24问题，一个典型的递归运算+暴力求解题，重要的是心细，能准确写出递归函数！也应该多去理解理解程序~本道题亮点之处在于判定两数是否相等时，由于是double运算，double型表示可能某些数无法准确表述，于是引入了从cmath库的fabs()函数用于计算两个数差值的绝对值，并定义一个double型极小数，用于辅助判别二者是否相等~

* **2801**

题目本身难度不大，但是看起来复杂，不易理解。可以了解的是对于字符串以及字符char型与int型的转化。

* **2806**

给出两个字符串，给出最长公共子字符串包含的字符数目

最长公共子序列是典型的套路题，主要还是其中的递归形式要理解，这是常规的套路，还是要熟悉熟悉！！

dp[i][j]表示字符串1的前i个字符和字符串2的前j个字符的最长公共子序列的长度

dp[0][j]=0

dp[i][0]=0

dp[i][j]=dp[i-1][j-1]+1 |{ str1[i]=str2[j] }

dp[i][j]=max(dp[i][j-1],dp[i-1][j]) |{ str1[i]!=str2[j] }

* **2808**

用数组去解决此问题，用空间换时间，难度不大。

* **2809**

2的高次幂求解，高精度运算，应用了快速幂思想，就是将p=a0 \* (2^0)+a1 \* (2^1)+a2 \*(2^2)+…..+an\*(2^n) 按此种形式拆分开，接下来求2^p=2^( a0 \* (2^0))\*2^( a1 \* (2^1))\*…\*2^( an\*(2^n)) 会发现彼此的相邻的乘积项满足：2^(2^(n-1)\* 2^(2^(n-1)=2^(2^n) 想当于 An\*An=An+1，这种易于求解，利用高精度乘法去循环求解即可。当然本题中提到N<=100，粗略估计2的幂次项顶多40多位，本道题在自己求解时直接调用麦森数模块写得，所以将结果以及乘积项都开到了500位，所以解此道题措措有余，关于求解，此处为了加快运算采用了十万进制，对于数组也是long long型数组开的，其中的循环书写难度不大~当然高精度乘法运算处碰了些小问题也是有些尴尬…一定要细心，把握住逻辑思路~在输出打印数据时，注意高位的多余的0剔除，以及首位五位数字不可进行高位补零操作！

本道题调用了cstring库的memset()对于数组初始化以及memcpy()对于数组复制，iomanip库的setfill()、setw()函数对于十万进制下的数高位进行补零操作，

* **2810**

完美立方问题，遍历所有的立方值，找到满足a3 = b3 + c3 + d3 的元素并按格式打印出来，整体来说是一道简单题~

* **2811**

熄灯问题，此道题其实是利用了题目中给的一个性质，就是每一行的下一行只有按下上一行亮的灯的按钮，才能使上一行灯灭，所以思路就是枚举出第一行的灯的按钮情况，只要第一行按钮可以使最终第五行的灯全灭，那么解出对应的方案。在此道题中，暴露了一些问题，本来思路很正确，但是解题特别慢，程序总是不严谨！对于数组描述的题型，一定要保证程序书写的正确性！关于数组的初始化，复制操作这些一定要思路严谨~本道题中用到了cstring库中的memcpy()用于数组的复制以及memset()用于数组的初始化，对于二维数组的初始化，需要注意的是memset(press,0,sizeof(press));这样就可以完成对press[]数组的初始化~

* **2812**

恼人的青蛙，wa了很多次，很恶心的一道题，对于一些剪枝条件判别得很不好，最重要的是要把住题目的关键，判别好条件！枚举+剪枝思路，详细的讲解在程序分析中有提到，此处不再多叙~用到了algorithm的qsort()库对于结构体struct数组进行快排，关于compare函数的书写，可以多去体会，掌握书写的方法~此外还用到了bsearch()函数对于元素的查询用二分查询，二分查询的前提要求数组元素已经排好序，所以快排+二分查找是查询元素的一个很好的方法~应多去了解bsearch()函数的用法！bsearch(&p,plant,N,sizeof(Plant),compare)，此处表示 元素p是查询对象，plant是被查询的数组，N是元素个数，sizeof(Plant)是元素的大小，compare是比较函数。

* **2816**

红与黑，地砖，其实就是图的广搜问题，题目较为简单，注意图中的节点被搜索后一定要打上标识~

* **2950**

摘花生问题，难度不大，分析好逻辑即可，模拟类型题，注意此处的绝对值函数abs()需要导入cmath库。

* **2964**

日历问题，给定天数，判别相应的年月日 周几，关于年月日的计算采用了迭代思路，一直减至相应的年份、月份，剩下的便是当月的日期，思路较为巧妙~

* **2965**

玛雅历题，此题细节处较多，题不难但坑点多！WA目前为止最多次数！留意的是取余时的一些细节的处理！

* **2966**

题目难度不大，但是题目需要思索的点很多…本道题中因为为了匹配字符串，同时还要得到相应的值，与是开了两个数组去解决本问题。此外应注意输入输出流控制形式的书写！引入<iomanip>库之后，setfill()以及setw()函数的调用~

* **2972**

进制转换，题目难度不大，注意不要犯变量初始化未附上初值便使用的这种低级错误，其次是可能数值较大，要换数制，不能只是int型存储！

* **2973**

题目较为简单，对于基本的数学公式的程序实现，主要需要留意的是，由于输入的数据值极大，只能采用字符串存储，以及strlen()是返回当前字符串长度的函数。

* **2974**

难度较大，主要是题目需要完成的点较多，留意调用algorithm库中的qsort()函数进行排序，同时关于字符串处理的strcmp函数也需要知道如何使用，并且一定要有头文件<string.h>或<cstring>！！

* **2975**

本道题难度较小，主要留意字符串的接受用cin.getline()函数，其余的解题思路比较正常，同时对于strcmp字符串比对函数需要会使用，该函数被使用的频率较高！

* **2977**

计算人的生理周期题，难度不大，就是反复枚举出满足条件的数，不断筛选，只是题目有一些细节的地方有些坑，需要注意，就是计算的时间一定是 d+1~d+21252之间的日期，这一处有些坑。

* **2980**

大整数乘法，高精度运算题，与大整数加法其实处理方式很类似，指定位相乘得到指定位的乘积，此处专门开一个大数组用于存储每位乘积，然后对于各位去判别是否产生进位。本道题用到了algorithm库中的reverse()函数，整体来说，题目如果分析清楚了，难度不是很大~

* **2981**

**解法一：**

大整数加法，高精度运算题，采取的方法通常为数组存储元素，模拟人手算的过程去计算最后的值，本道题中用到了algorithm库里的max()函数，以及cstring库里的 fill()函数，对于fill()函数的使用要多留意，再次用到了reverse()函数用于对字符串数组的逆序。本道题的细节问题较多，比如数组开大一些，以防产生进位导致数组溢出；对于数组中的陷阱：高位0应该除去以及倘若均为 0+0时又不能再除去0这些小细节的问题应多加留意，以防犯错~

**解法二：**

大整数加法，高精度运算题，解法二较之解法一可能思路上更死板，但是更容易编出代码，就是和2980的大整数乘法的解法思路基本一致~

* **2996**

选课问题，水题，就是获取当前序列的下M个序列，真正自己去撸代码可能难度不小，但是可以直接调用algorithm库的next\_permutation()函数，直接AC~

* **4001**

很套路的广搜问题，本质上就是基础的考查，但是本题需要谨记的是因为一些思维的不严谨，疯狂wa！！最后发现是队列没有清空以及初始点没有标记为true，要长记性了！！

* **4004**

有n个正整数，找出其中和为t(t也是正整数)的可能的组合方式。

利用深搜划分枚举所有的子集合，求出使得组合值为t的集合数目。关于子集合的划分有一定的套路可寻，本道题比较具有典型性，可以思考思考~

* **4079**

二叉搜索树的构造以及前序遍历，是典型套路题，关于程序如何书写以及如何做到函数递归调用都需要了然于胸！要清楚题目该从哪个套路点出发去解~

* **4007**

本道题是编辑距离问题，然而我感觉AC的代码有问题，此处是我觉得正确但是WA的代码思路：

首先定义这样一个函数——edit(i, j)，它表示第一个字符串的长度为i的子串到第二个字符串的长度为j的子串的编辑距离。

显然可以有如下动态规划公式。

if i == 0 且 j == 0，edit(i, j) = 0

if i == 0 且 j > 0，edit(i, j) = j

if i > 0 且j == 0，edit(i, j) = i

if i ≥ 1 且 j ≥ 1 ，edit(i, j) == min{ edit(i-1, j) + 1, edit(i, j-1) + 1, edit(i-1, j-1) + f(i, j) }，

当第一个字符串的第i个字符不等于第二个字符串的第j个字符时，f(i, j) = 1；否则，f(i, j) = 0。

* **4082**

二叉树的镜面映射，是一道纯粹考查编程实现能力的二叉树问题，难度较大不易处理，本道题是结合网上的理解勉强写出来的程序...之后可以加强理解得去刷此题~

* **4084**

典型的拓扑排序问题，主要是先要知道拓扑排序：拓扑排序，就是每次找到当前入度为0的节点将它弹出，然后连带着它所指向的边也都删除，然后再找下一个入度为0的节点，直至整张图遍历完。

因为本题是要求每次都是编号小的顶点弹出，于是构造一个优先队列，每次队头存放的都是最小值。拓扑排序有套路的模板去构造，对于这种模板题，一定要有独立编写出的能力！！

* **4102**

典型的二维背包问题，因为本道题有两个约束条件，一个最优目标，一开始试图套一维 0-1背包模板，发现思路打不开，后来从二维背包问题思索，开二维数组去解此问，问题成功求解，具体套路实际上和0-1背包思路基本一致！

* **4108**

羚羊增长问题，可以手推递归公式，难度不大。

* **4115**

广搜问题，题目有些小陷阱，就是查克拉也属于一维的变量，因为不同的查克拉到达地图同一点的意义也是不一样的，所以应在三维层面上去思考本问题~

* **4117**

简单的整数划分问题，本道题仍然用递归思路去分析：

当 m = 1 时，不论k的值为多少（k > 0 )，只有一种划分即 { 1 };

当 n = 1 时，不论m的值为多少，只有一种划分即 m 个 1，{ 1, 1, 1, ..., 1 };

当 m = n 时，根据划分中是否包含 n，可以分为两种情况：

(a)划分中包含n的情况，只有一个即 { n }；

(b)划分中不包含n的情况，这时划分中最大的数字也一定比 n 小，即 n 的所有 ( n- 1 ) 划分。因此 f(m, n) = 1 + f(n, n-1);

当 m < n 时，由于划分中不可能出现负数，因此就相当于 f(m, m);

当 m > n 时，根据划分中是否包含最大值 n，可以分为两种情况：

(a)划分中包含 n的情况，划分个数为 f(m-n,n);

(b)划分中不包含 n 的情况，则划分个数为 f(m,n-1);因此 f(m, n) = f(m-n, n) + f(m, n-1);

按照这种递归的思想，即可解出题目，不过这种思路一定要多加了解，至今尚且不是很

明白递归的思想…

* **4131**

典型的0-1背包问题，题目主要是要理解动归思路~难度不大，模板题型！

* **4132**

四则运算表达式求值，是一个蛮复杂的问题，考虑的点比较多，此处的题解思路参照网上的博客：<https://blog.csdn.net/sgbfblog/article/details/8001651> 就是对于四则表达式将其转换为易于用计算机语言处理的逆波兰表达式，对于中缀表达式转换为后缀表达式，有标准的转换方法：

eg: a + b\*c + (d \* e + f) \* g 🡪 a b c \* + d e \* f + g \* + ；

后缀表达式又被称为逆波兰表达式，从左至右，表达式每个元素依次入栈，对于数值型直接进栈，对于符号，则是对应的当前栈顶元素与次栈顶元素按照符号运算法则运算。

故本道题的核心是如何将前缀表达式转换为后缀表达式，简单来说，就是对于数值型的直接输出，而关于符号型，需要比较优先级后输出，博客中给出了如何转换的图示，但是编程实现时还需要利用栈结构以及字符数组去存储输出的逆波兰表达式，同时本处为了利于之后的double型数字提取，在数字型变量后面添加了空格符以作为隔开的标志，最后获得了转换后的数组之后，再按照逆波兰表达式求法利用栈结构进行求值。

本道题主要利用了主要利用了stack库的一系列入栈、出栈、检测非空、取栈顶值等函数；iomanip库的setprecision()以及fixed()函数用于设置输出的小数点精度；cstring的memset()函数用于初始化数组；cmath库的atof()函数用于将字符数组转换为相应的double型变量。

关于本道题，因为四则运算表达式求值是一个经典问题，可以反复琢磨直至熟悉解法。

* **4145**

本道题是典型的求最大平均值问题，题目的思路较为刁钻，只有做过类似的题目，才有可能相出合适的题解~

本道题其实是找 ∑vi / ∑wi >=x的最大x，把不等式进行变形，就得到了 ∑ ( vi - x \* wi )>=0，于是判断就成了，对 vi-x\*wi 的值进行排序之后贪心进行选择，判断去除k个最小值后是否还大于等于0。对于x的寻找是用了二分查找，由于都是double型变量通常关于double型变量的比对，一般是设置一个极小值，比如eps=1e-6，判断两个数大小或者是判断数是否等于零这一类的问题时都是用种方法去比对，然后再四舍五入的表述时，我用了cmath库的ceil()和floor()函数去比对，可能方法不是最优，但很好地去完成了四舍五入的要求。本道题wa了五次，其实思路从第二次就走对了，但是始终会有些小问题也是蛮尴尬的，最主要的还是compare的函数书写，不知道为什么，用另一种方法书写时就会出现奇怪的问题..本道题的compare()函数的书写应该是写得比较不一样的一次，值得反复研读~

* **9284**

盒子与小球问题之二

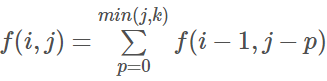
盒子与球的投放问题，看网上题解感觉是指每个盒子最多放两颗球:A和B；或者可以仅某种球一颗，或者不放球~

就是简单的组合问题，将A个相同的球放入N个不同的盒子中，每个盒子可以放一颗A球或者不放，其实就相当于 将A个球补上N个空球，然后选出N个球与盒子匹配上

本道题 主要注意阶层计算时数值会溢出的问题，如果图方便的话可以直接写一个大数乘法，此处是用long long 型长类型数据存储解决~

* **9285**

盒子与小球问题之三

本道题是一道动态规划问题，从最基本的动态规划递推式导出 出发：思路就是dp[i][j]来表示前i个盒子中有j个球，那么这种可能的数目其实是对于前i-1个盒子中有 j-k~j个球的概率，当然有个附加条件，就是 此处的求和因为可能本身j都小于k，所以要有个额外的小处理。

由此递推公式可以写出三层循环用于求解，但是因为数据量较多，所以会TL，这是为了减少程序运行时间，需要引入前缀和公式去减少求解时间：

f[i][j]=f[i-1][j]+f[i][j-1]-f[i-1][j-k-1]; //当j>k时

依据此公式可以节省程序运行时间，少了一重循环，但是我的代码有些bug，所以只能跑通8个样例…

* **9286**

盒子与小球问题之四

一开始用纯粹的组合数学的思路去想这个问题，发现思路行不通！！

本道题和poj 上的 Push Botton Lock的题目区别是，它是 n个不同的球投入m个不同的盒子中同时每个盒子至少含有K个球，而 poj 上的 Push Botton Lock 是 n个不同的球投入m个不同的盒子中同时每个盒子至少含有1个球。相较于 poj上的 Push Botton Lock 题目是典型的第二类斯特灵数的求法，本道题是有些许改进，但实际上只是 (N,M,K)特定情况下的方案数 本道题比较经典，建议多多研读理解！

f[i][j]=sum(f[i-k][j-1]\*c[n-(i-k)][k])-- k: K~i

* **2016年计算所机试上机前训练 D题**

给出一个数学方式构造出一个二维层次的数字矩阵，细细地分析列举出所有的情况即可~